Chapter 12

Cryptographic Hash Functions

Objectives

- To introduce general ideas behind cryptographic hash functions
- To discuss the Merkle-Damgard scheme as the basis for iterated hash functions
- To distinguish between two categories of hash functions:
- To discuss the structure of SHA-512.
- To discuss the structure of Whirlpool.

12-1 INTRODUCTION

- ***A cryptographic hash function takes a message of arbitrary length and creates a message digest of fixed length.***
- ***Instead of creating a hash function with variable size input, hash function with fixed size input will be created and it will be used required number of time.***
- ***The fixed size input function is known as compression function.***
- ***The Scheme is referred as iterated cryptographic hash function.***

12.1.1 Iterated Hash Function

Merkle-Damgard Scheme

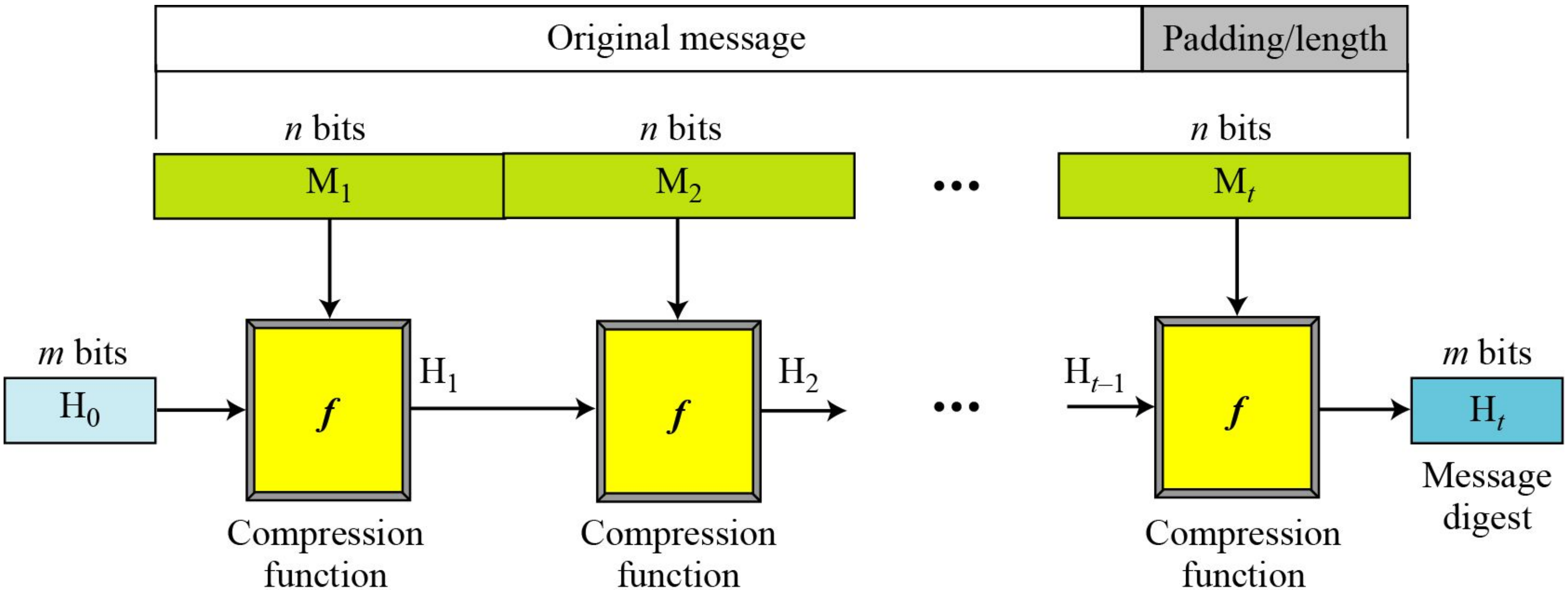


Figure 12.1 ***Merkle-Damgard scheme***

12.1.2 Two Groups of Compression Functions

1. The compression function is made from scratch.

Message Digest (MD)
Secure Hash Algorithm(SHA)
RIPEMD-160

2. A symmetric-key block cipher serves as a compression function.

Whirlpool

Hash Function Made from Scratch

Message Digest(MD)

- Designed by Ron Rivest
- versions- MD2,MD4,MD5
- MD5 divided msg into blocks of 512 bits and creates 128-bit digest which is too small to resist collision attack.

Secure Hash Function(SHA)

- Based on MD5
- Versions- SHA-128, SHA-224, SHA-256, SHA-384 and SHA-512

Other Algorithms

- RACE Integrity Primitives Evaluation message Digest(RIPMED)
- RIPMED-160 is based on MD5 but two line of parallel execution is there.
- HAVAL is variable length hashing algorithm with message digest size 128, 160, 192, 224 and 256 where block size is 1024.

Comparison of SHA

Table 12.1 *Characteristics of Secure Hash Algorithms (SHAs)*

Characteristics	*SHA-1*	*SHA-224*	*SHA-256*	*SHA-384*	*SHA-512*
Maximum Message size	$2^{64} - 1$	$2^{64} - 1$	$2^{64} - 1$	$2^{128} - 1$	$2^{128} - 1$
Block size	512	512	512	1024	1024
Message digest size	160	224	256	384	512
Number of rounds	80	64	64	80	80
Word size	32	32	32	64	64

Rabin Scheme

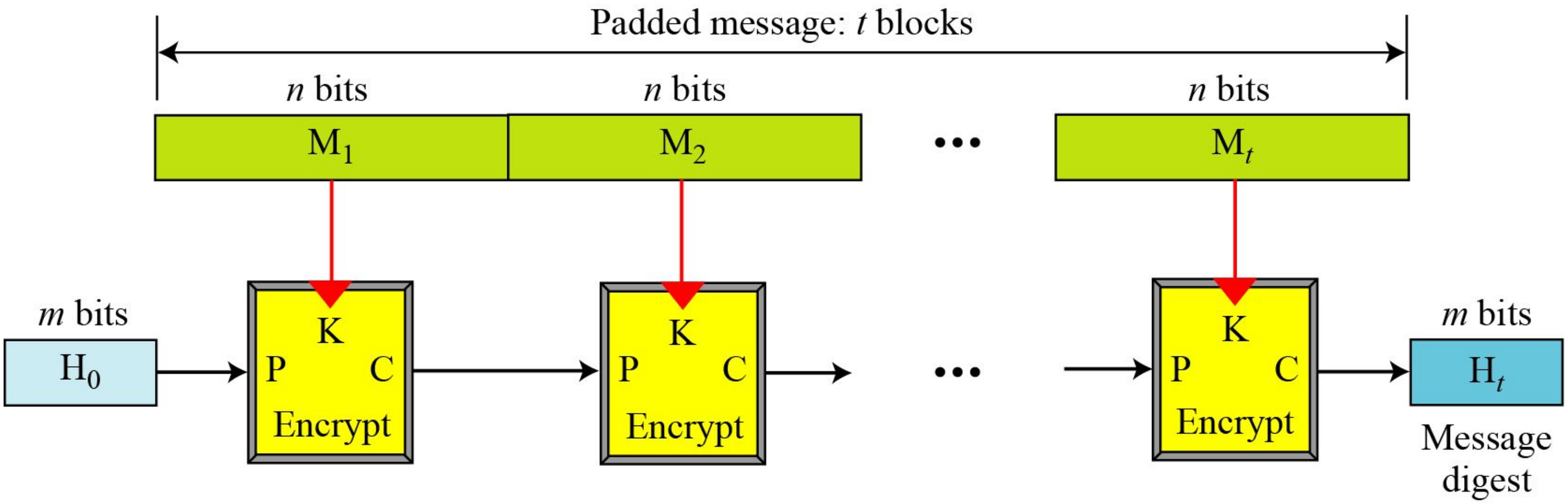


Figure 12.2 ***Rabin scheme***

Davies-Meyer Scheme

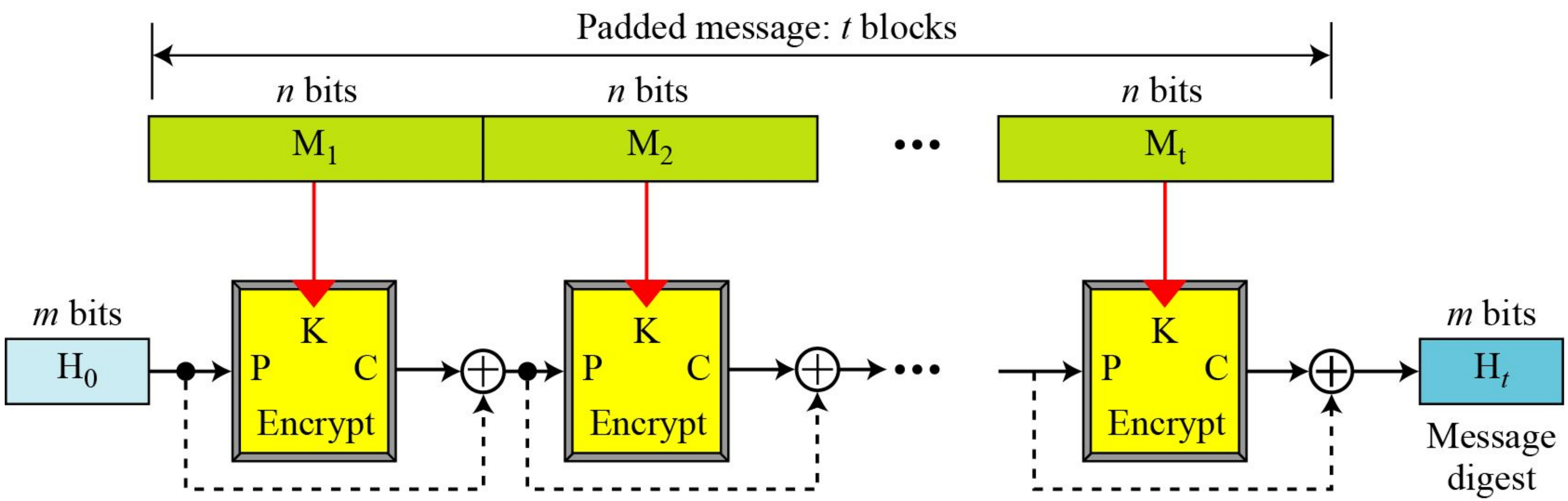


Figure 12.3 ***Davies-Meyer scheme***

Matyas-Meyer-Oseas Scheme

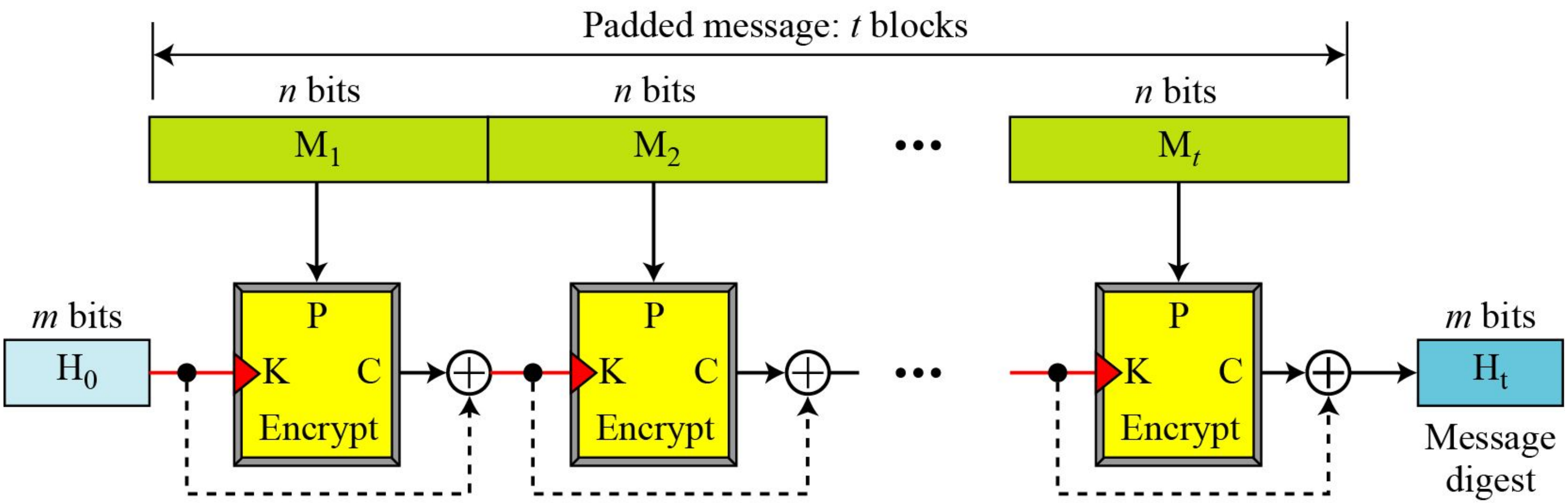


Figure 12.4 ***Matyas-Meyer-Oseas scheme***

12.1.2 Continued

Miyaguchi-Preneel Scheme

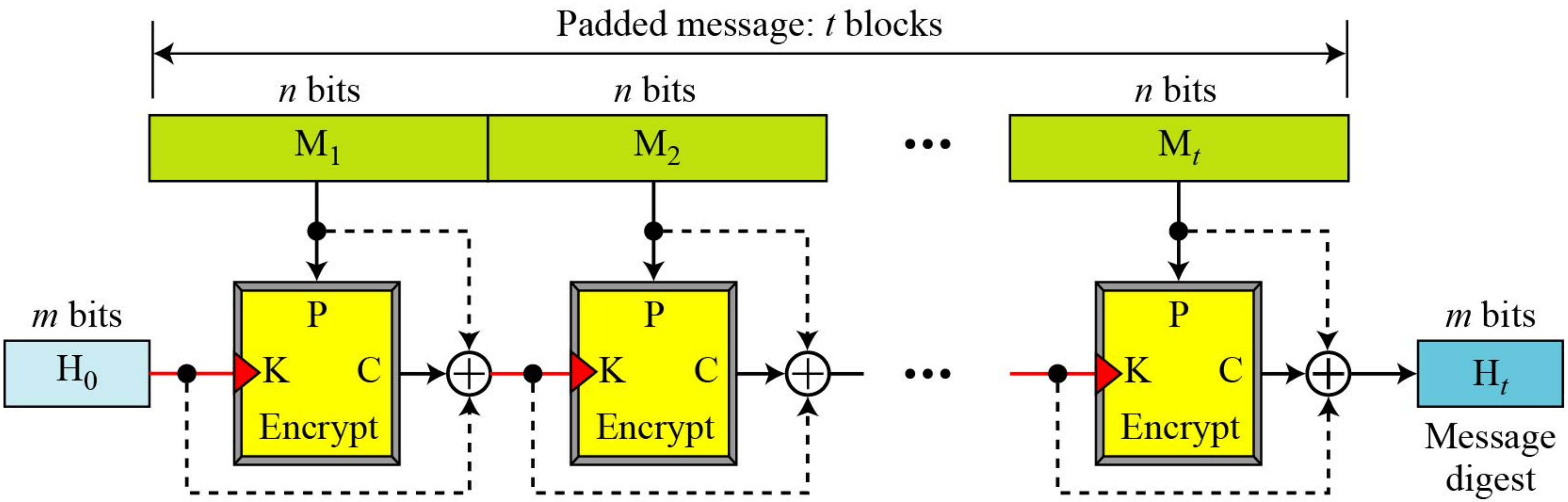


Figure 12.5 *Miyaguchi-Preneel scheme*

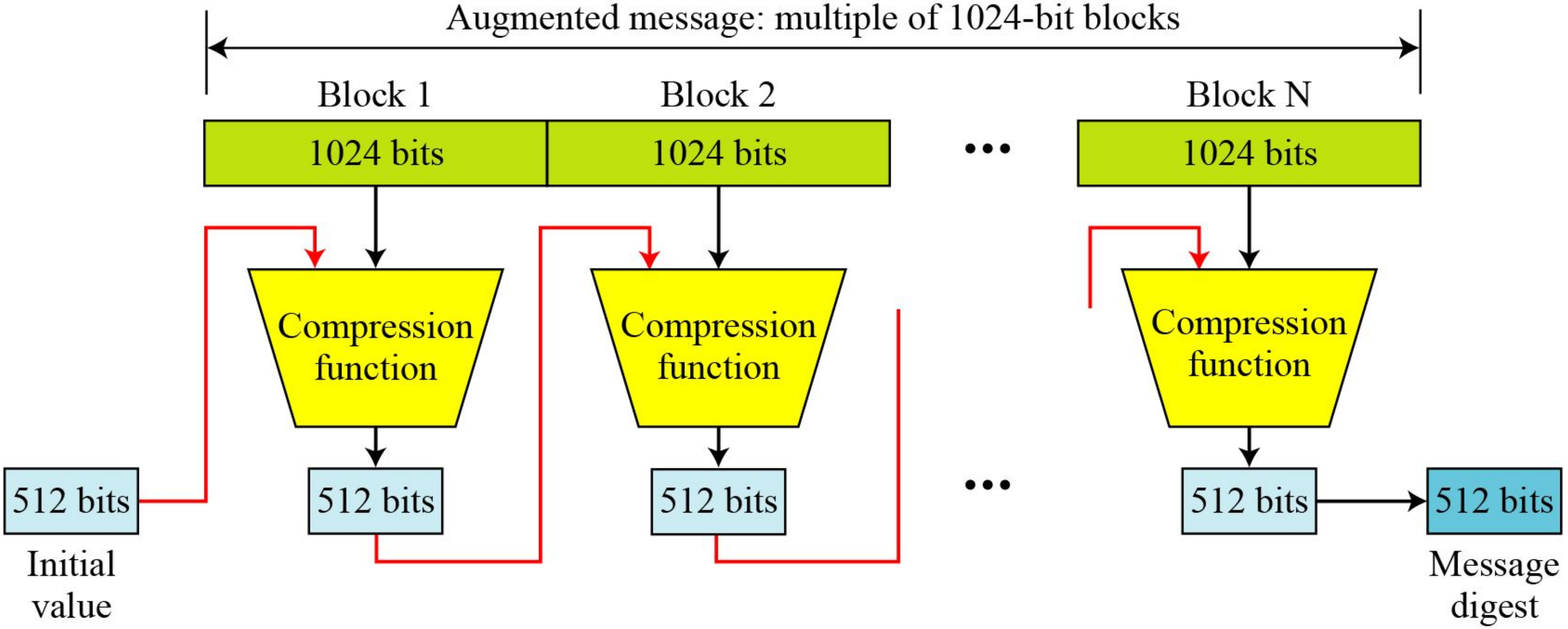
12-2 SHA-512

SHA-512 is the version of SHA with a 512-bit message digest. This version, like the others in the SHA family of algorithms, is based on the Merkle-Damgard scheme.

Topics discussed in this section:

12.2.1 Introduction
12.2.2 Compression Function
12.2.3 Analysis

Figure 12.6 *Message digest creation SHA-512*



Message Preparation

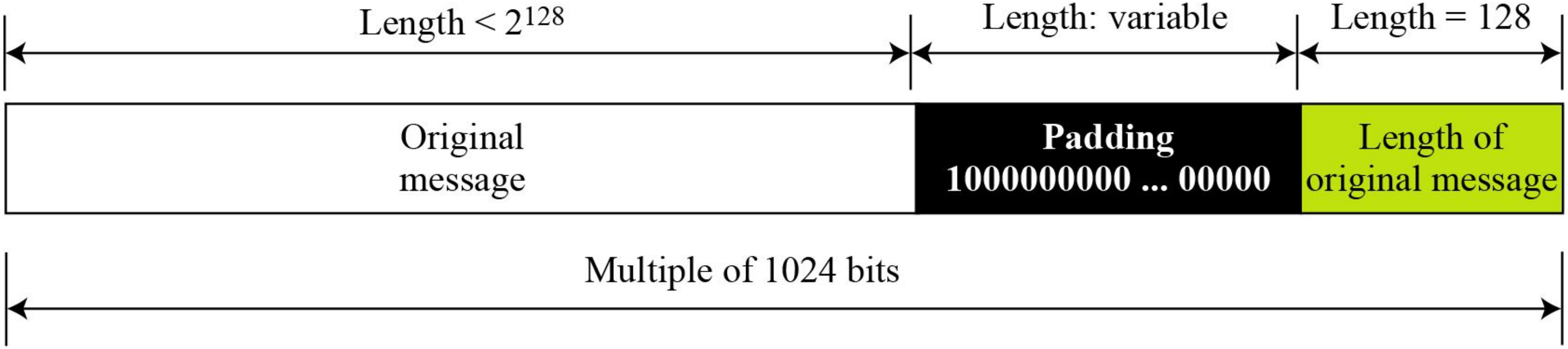
SHA-512 insists that the length of the original message be less than 2^{128} bits.

Note

SHA-512 creates a 512-bit message digest out of a message less than 2^{128}.

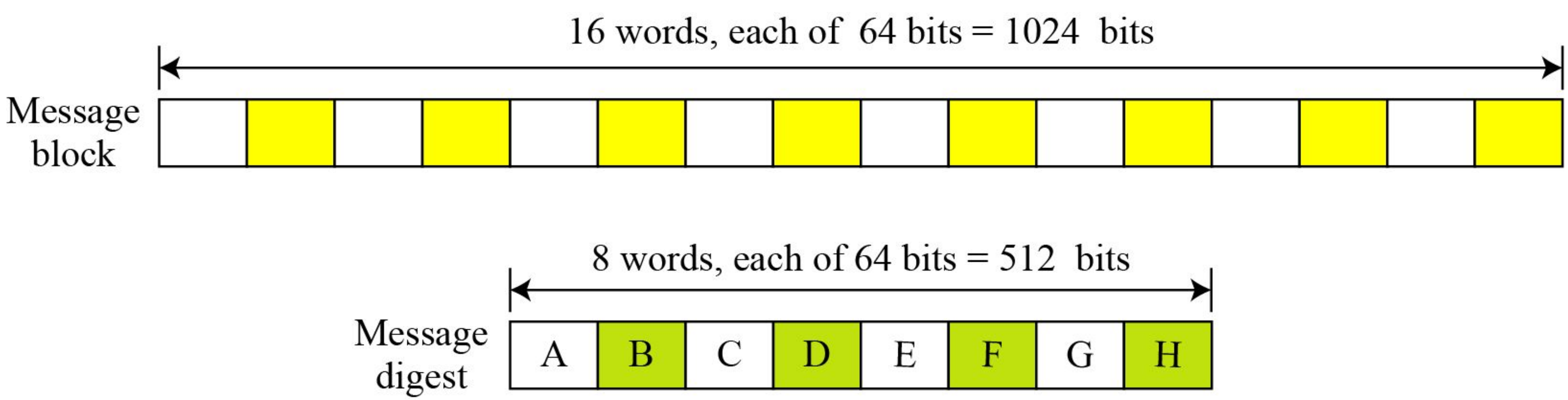
12.2.1 Continued

Figure 12.7 ***Padding and length field in SHA-512***



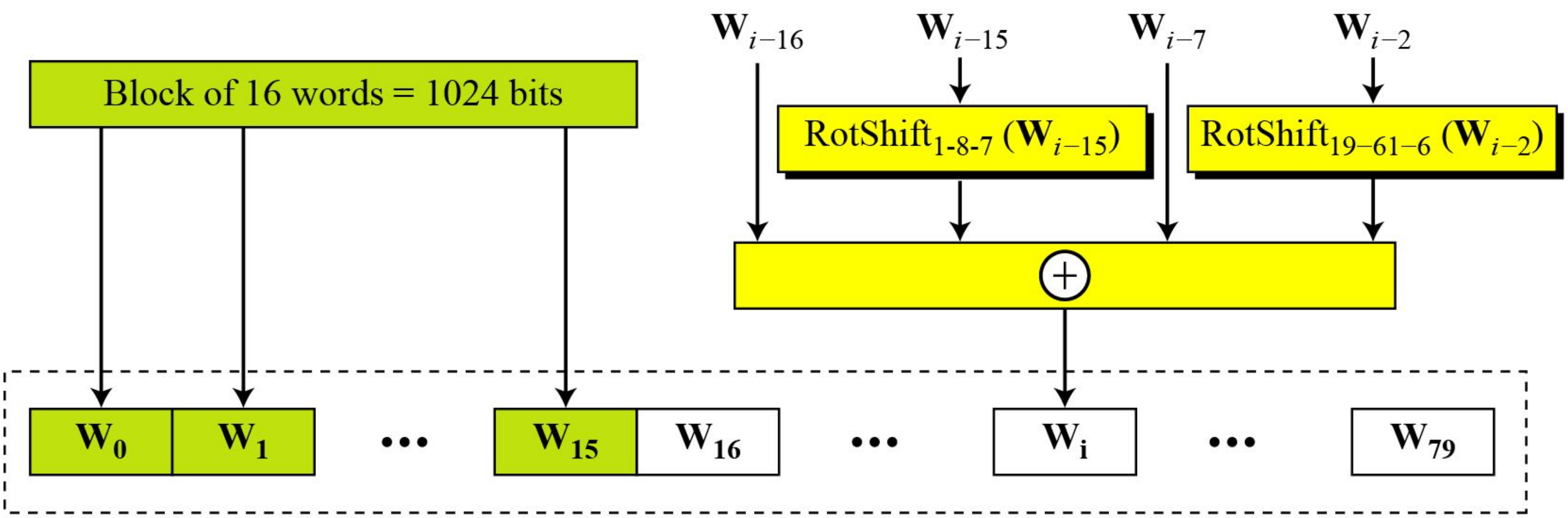
Words

Figure 12.8 *A message block and the digest as words*



Word Expansion

Figure 12.9 *Word expansion in SHA-512*



$\text{RotShift}_{l\text{-}m\text{-}n}(x)$: $\text{RotR}_l(x) \oplus \text{RotR}_m(x) \oplus \text{ShL}_n(x)$

$\text{RotR}_i(x)$: Right-rotation of the argument x by i bits
$\text{ShL}_i(x)$: Shift-left of the argument x by i bits and padding the left by 0's.

Message Digest Initialization

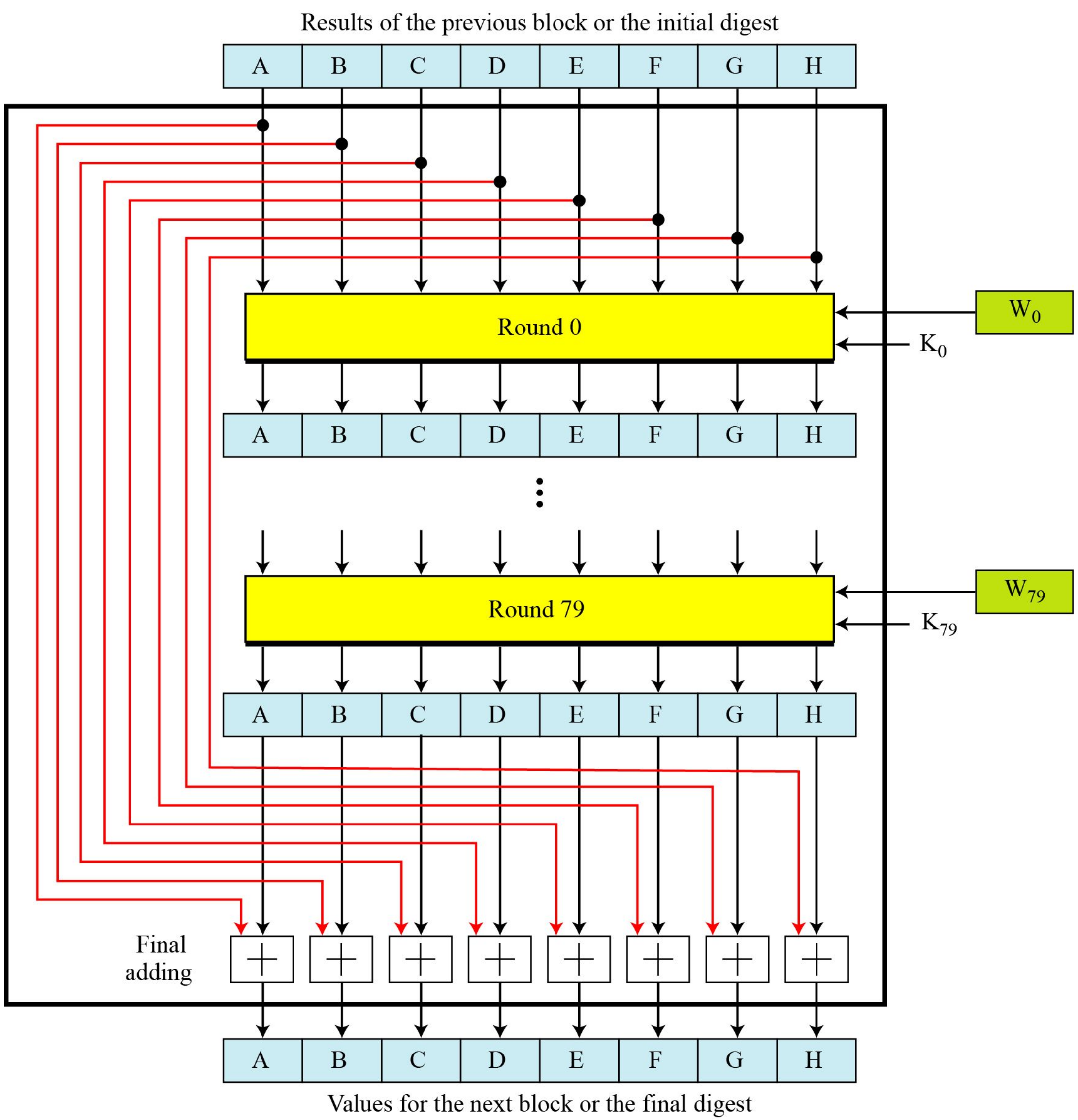
Table 12.2 *Values of constants in message digest initialization of SHA-512*

Buffer	*Value (in hexadecimal)*	*Buffer*	*Value (in hexadecimal)*
A_0	**6A09E667F3BCC908**	E_0	**510E527FADE682D1**
B_0	**BB67AE8584CAA73B**	F_0	**9B05688C2B3E6C1F**
C_0	**3C6EF372EF94F828**	G_0	**1F83D9ABFB41BD6B**
D_0	**A54FE53A5F1D36F1**	H_0	$\boxed{\textbf{5BE0CD19137E2179}}$

- ***Each Value is a fraction part of the square root of the corresponding prime after converting to the binary and keeping only the 64 bits.***
- ***Ex: 8^{th} prime=19***
- ***sqrt(19)= 4.35889894354□$(100.0101 \ldots 1001)_2$***
  $(4.\ 5B\boxed{E0CD19137E2179})_{16}$

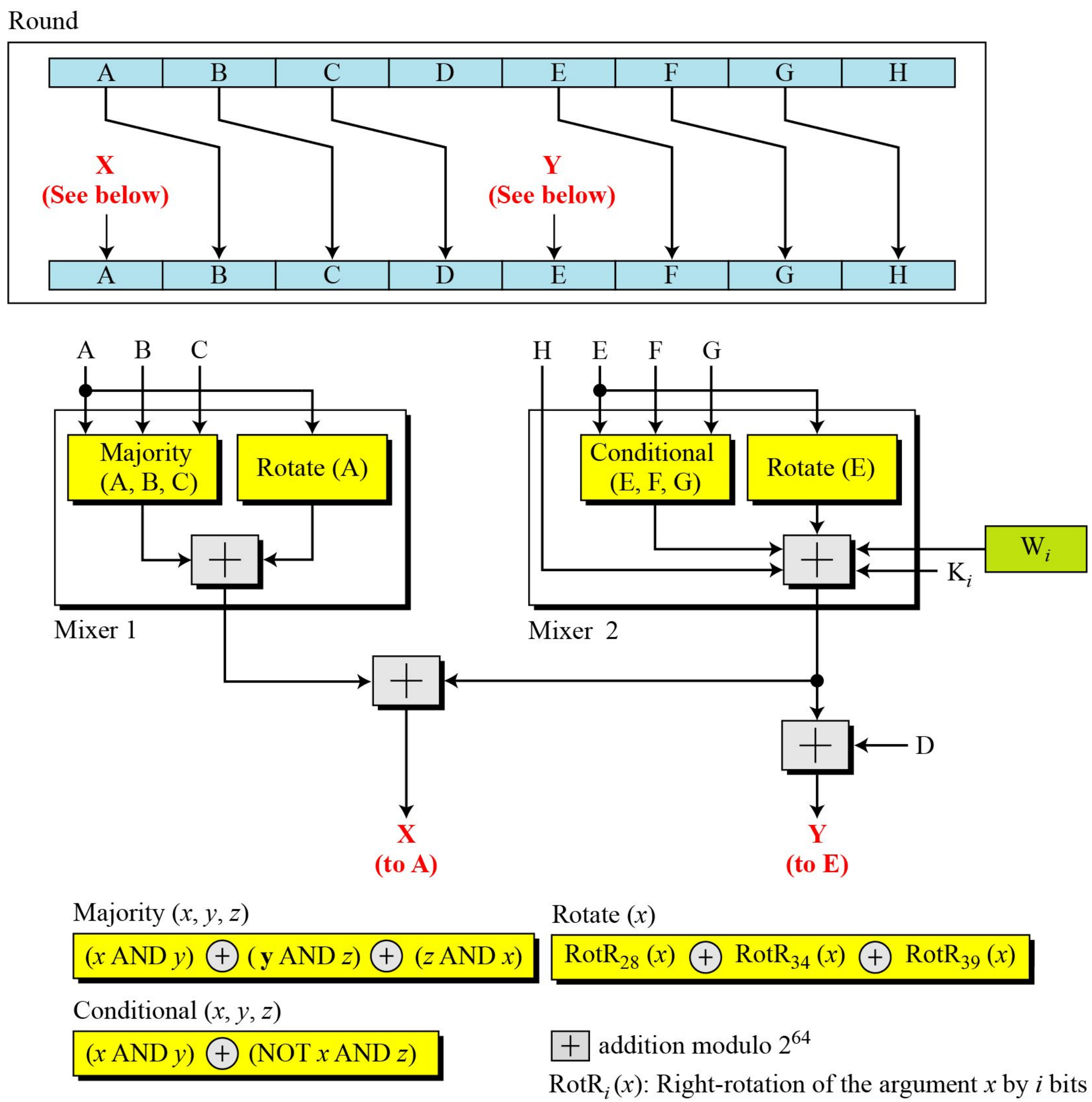
12.2.2 Compression Function

Figure 12.10 *Compression function in SHA-512*



12.2.2 Continued

Figure 12.11 *Structure of each round in SHA-512*



12.2.2 Continued

Majority Function

$$(A_j \text{ AND } B_j) \oplus (B_j \text{ AND } C_j) \oplus (C_j \text{ AND } A_j)$$

Conditional Function

$$(E_j \text{ AND } F_j) \oplus (\text{NOT } E_j \text{ AND } G_j)$$

Rotate Functions

$$\text{Rotate (A): } \text{RotR}_{28}(A) \oplus \text{RotR}_{34}(A) \oplus \text{RotR}_{29}(A)$$

$$\text{Rotate (E): } \text{RotR}_{28}(E) \oplus \text{RotR}_{34}(E) \oplus \text{RotR}_{29}(E)$$

12.2.2 Continued

Table 12.3 *Eighty constants used for eighty rounds in SHA-512*

428A2F98D728AE22	7137449123EF65CD	B5C0FBCFEC4D3B2F	E9B5DBA58189DBBC
3956C25BF348B538	59F111F1B605D019	923F82A4AF194F9B	AB1C5ED5DA6D8118
D807AA98A3030242	12835B0145706FBE	243185BE4EE4B28C	550C7DC3D5FFB4E2
72BE5D74F27B896F	80DEB1FE3B1696B1	9BDC06A725C71235	C19BF174CF692694
E49B69C19EF14AD2	EFBE4786384F25E3	0FC19DC68B8CD5B5	240CA1CC77AC9C65
2DE92C6F592B0275	4A7484AA6EA6E483	5CB0A9DCBD41FBD4	76F988DA831153B5
983E5152EE66DFAB	A831C66D2DB43210	B00327C898FB213F	BF597FC7BEEF0EE4
C6E00BF33DA88FC2	D5A79147930AA725	06CA6351E003826F	142929670A0E6E70
27B70A8546D22FFC	2E1B21385C26C926	4D2C6DFC5AC42AED	53380D139D95B3DF
650A73548BAF63DE	766A0ABB3C77B2A8	81C2C92E47EDAEE6	92722C851482353B
A2BFE8A14CF10364	A81A664BBC423001	C24B8B70D0F89791	C76C51A30654BE30
D192E819D6EF5218	D69906245565A910	F40E35855771202A	106AA07032BBD1B8
19A4C116B8D2D0C8	1E376C085141AB53	2748774CDF8EEB99	34B0BCB5E19B48A8
391C0CB3C5C95A63	4ED8AA4AE3418ACB	5B9CCA4F7763E373	682E6FF3D6B2B8A3
748F82EE5DEFB2FC	78A5636F43172F60	84C87814A1F0AB72	8CC702081A6439EC
90BEFFFA23631E28	A4506CEBDE82BDE9	BEF9A3F7B2C67915	C67178F2E372532B
CA273ECEEA26619C	D186B8C721C0C207	EADA7DD6CDE0EB1E	F57D4F7FEE6ED178
06F067AA72176FBA	0A637DC5A2C898A6	113F9804BEF90DAE	1B710B35131C471B
28DB77F523047D84	32CAAB7B40C72493	3C9EBE0A15C9BEBC	431D67C49C100D4C
4CC5D4BECB3E42B6	4597F299CFC657E2	5FCB6FAB3AD6FAEC	6C44198C4A475817

12.2.2 Continued

There are 80 constants, K_0 to K_{79}, each of 64 bits. Similar These values are calculated from the first 80 prime numbers (2, 3,…, 409). For example, the 80th prime is 409, with the cubic root $(409)^{1/3}$ = 7.42291412044. Converting this number to binary with only 64 bits in the fraction part, we get

$$\boxed{(111.0110\ 1100\ 0100\ 0100 \ldots 0111)_2 \quad \rightarrow \quad (7.6C44198C4A475817)_{16}}$$

The fraction part: $(6C44198C4A475817)_{16}$

12.2.3 Analysis

With a message digest of 512 bits, SHA-512 expected to be resistant to all attacks, including collision attacks.

Questions

1. *Find the result of RotR12(x) if*
   X=1234 5678 ABCD 2345 34564 5678 ABCD 2468

2. *Find the result of ShL12(x) if*
   X=1234 5678 ABCD 2345 34564 5678 ABCD 2468

2. *Find the result of Rotate(x) if*
   X=1234 5678 ABCD 2345 34564 5678 ABCD 2468

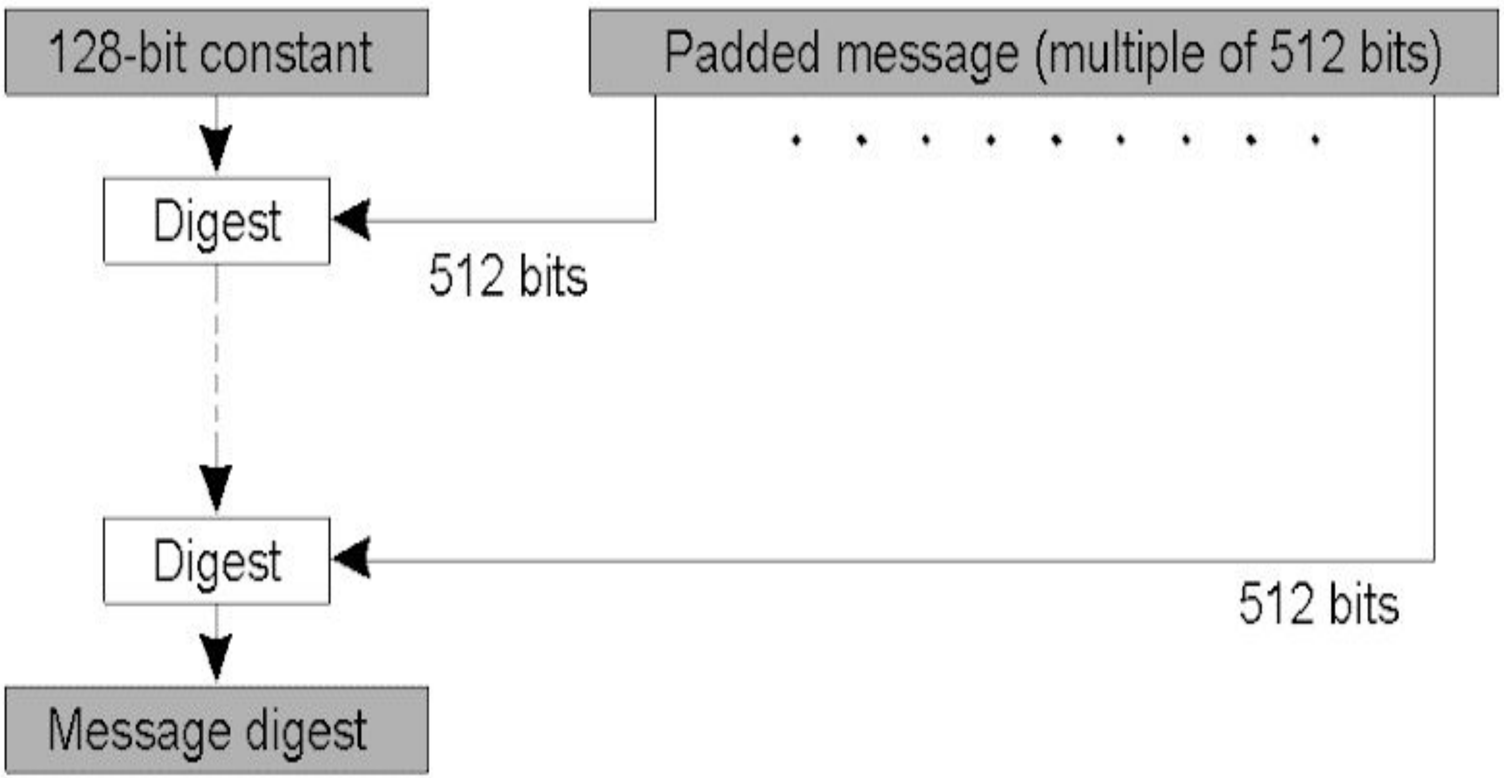
2. *Find the majority (x,y,z) and Conditional (x,y,z) if*
   X=1234 5678 ABCD 2345 34564 5678 ABCD 2468
   y=2234 5678 ABCD 2345 34564 5678 ABCD 2468
   z=3234 5678 ABCD 2345 34564 5678 ABCD 2468

MD Family

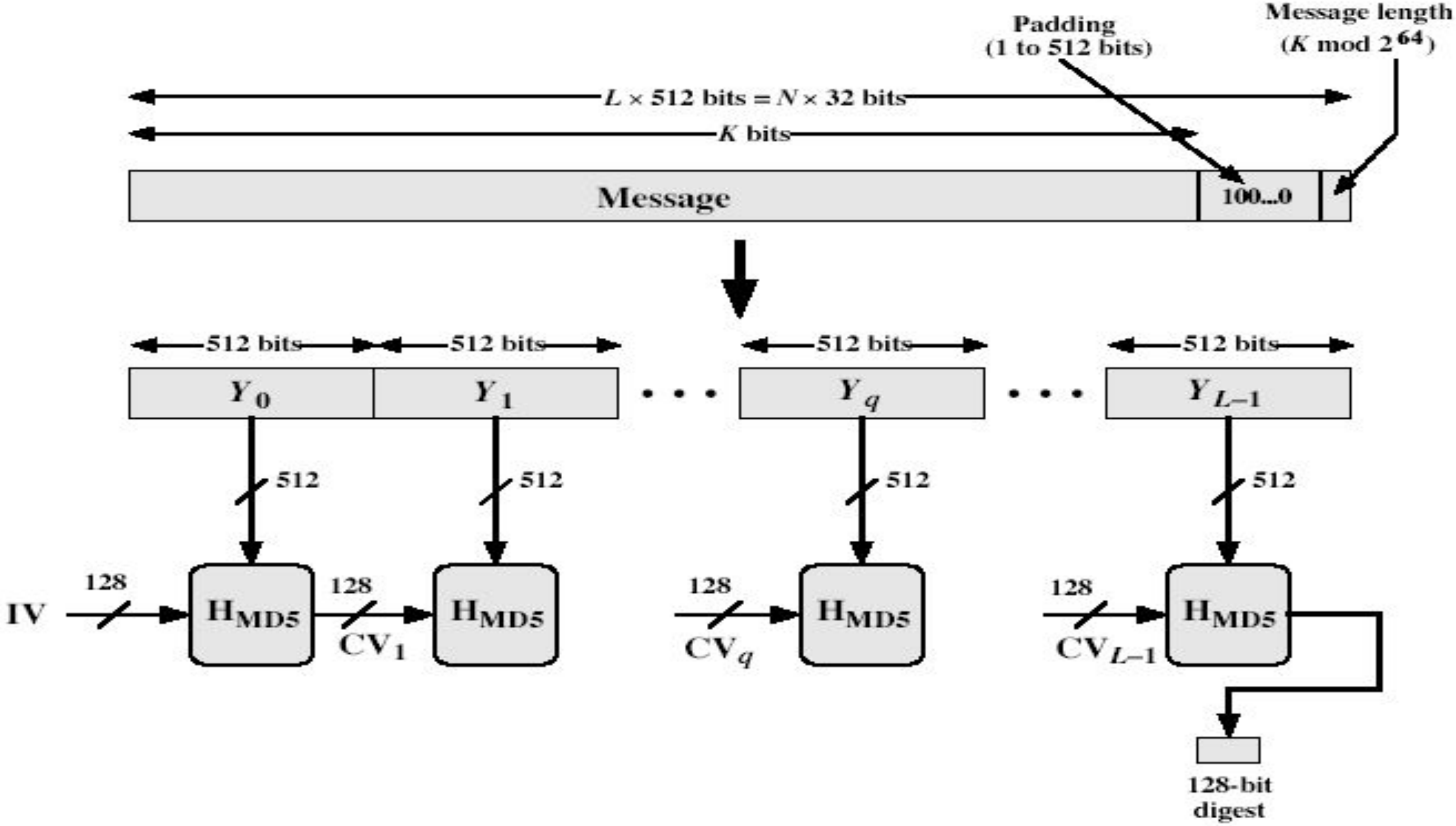
Introduction MD5

- Fifth iteration developed by Professor Ronald L. Rivest (RSA) in 1991.
- According to RFC 1321, “MD5 message-digest algorithm takes as input a message of arbitrary length and produces as output a 128-bit "fingerprint" or "message digest" of the input …
- The MD5 algorithm is intended for digital signature applications, where a large file must be "compressed" in a secure manner before being encrypted with a private (secret) key under a public-key cryptosystem such as RSA.”
- MD5 is optimized for use on 32-bit computers

MD5 Algorithm Structure



MD5 Algorithm



Implementation Steps

Step1 Append padding bits

- The input message is "padded" (extended) so that its length (in bits) equals to 448 mod 512.
- Padding is always performed, even if the length of the message is already 448 mod 512.
- Padding is performed as follows: a single "1" bit is appended to the message, and then "0" bits are appended so that the length in bits of the padded message becomes congruent to 448 mod 512.
- At least one bit and at most 512 bits are appended.

Implementation Steps

Step2. Append length

- A 64-bit representation of the length of the message is appended to the result of step1.
- If the length of the message is greater than 2^64, only the low-order 64 bits will be used.
- The resulting message (after padding with bits and with b) has a length that is an exact multiple of 512 bits.
- The input message will have a length that is an exact multiple of 16 (32-bit) words.

Implementation Steps

Step3. Initialize MD buffer

- A four-word buffer (A, B, C, D) is used to compute the message digest. Each of A, B, C, D is a 32-bit register. These registers are initialized to the following values in hexadecimal, low-order bytes first):

word A: 01 23 45 67

word B: 89 ab cd ef

word C: fe dc ba 98

word D: 76 54 32 10

Implementation Steps

Step 4 Process message in 16-word blocks

- Four functions will be defined such that each function takes an input of three 32-bit words and produces a 32-bit word output.

  F (X, Y, Z) = XY or not (X) Z
  G (X, Y, Z) = XZ or Y not (Z)
  H (X, Y, Z) = X xor Y xor Z
  I (X, Y, Z) = Y xor (X or not (Z))

- Each round processes the data using:
  - A circular left rotation (bitwise shift)
  - A constant table (T[i]), derived from

    $T[i] = \text{floor}(2^{32} \times \text{abs}(\sin(i)))$

Implementation Steps

Round 1.

[abcd k s i] denote the operation $a = b + ((a + F(b, c, d) + X[k] + T[i]) <<< s)$.

Do the following 16 operations.

[ABCD 0 7 1]	[DABC 1 12 2]	[CDAB 2 17 3]	[BCDA 3 22 4]
[ABCD 4 7 5]	[DABC 5 12 6]	[CDAB 6 17 7]	[BCDA 7 22 8]
[ABCD 8 7 9]	[DABC 9 12 10]	[CDAB 10 17 11]	[BCDA 11 22 12]
[ABCD 12 7 13]	[DABC 13 12 14]	[CDAB 14 17 15]	[BCDA 15 22 16]

MD5 vs. MD4

- A fourth round has been added.
- Each step has a unique additive constant.
- The function g in round 2 was changed from (XY v XZ v YZ) to (XZ v Y not(Z)).
- Each step adds in the result of the previous step.
- The order in which input words are accessed in rounds 2 and 3 is changed.
- The shift amounts in each round have been optimized. The shifts in different rounds are distinct.

Summary

- Comparing to other digest algorithms, MD5 is simple to implement, and provides a "fingerprint" or message digest of a message of arbitrary length.
- It performs very fast on 32-bit machine.
- MD5 is being used heavily from large corporations, such as IBM, Cisco Systems, to individual programmers.
- MD5 is considered one of the most efficient algorithms currently available.

Thank You…